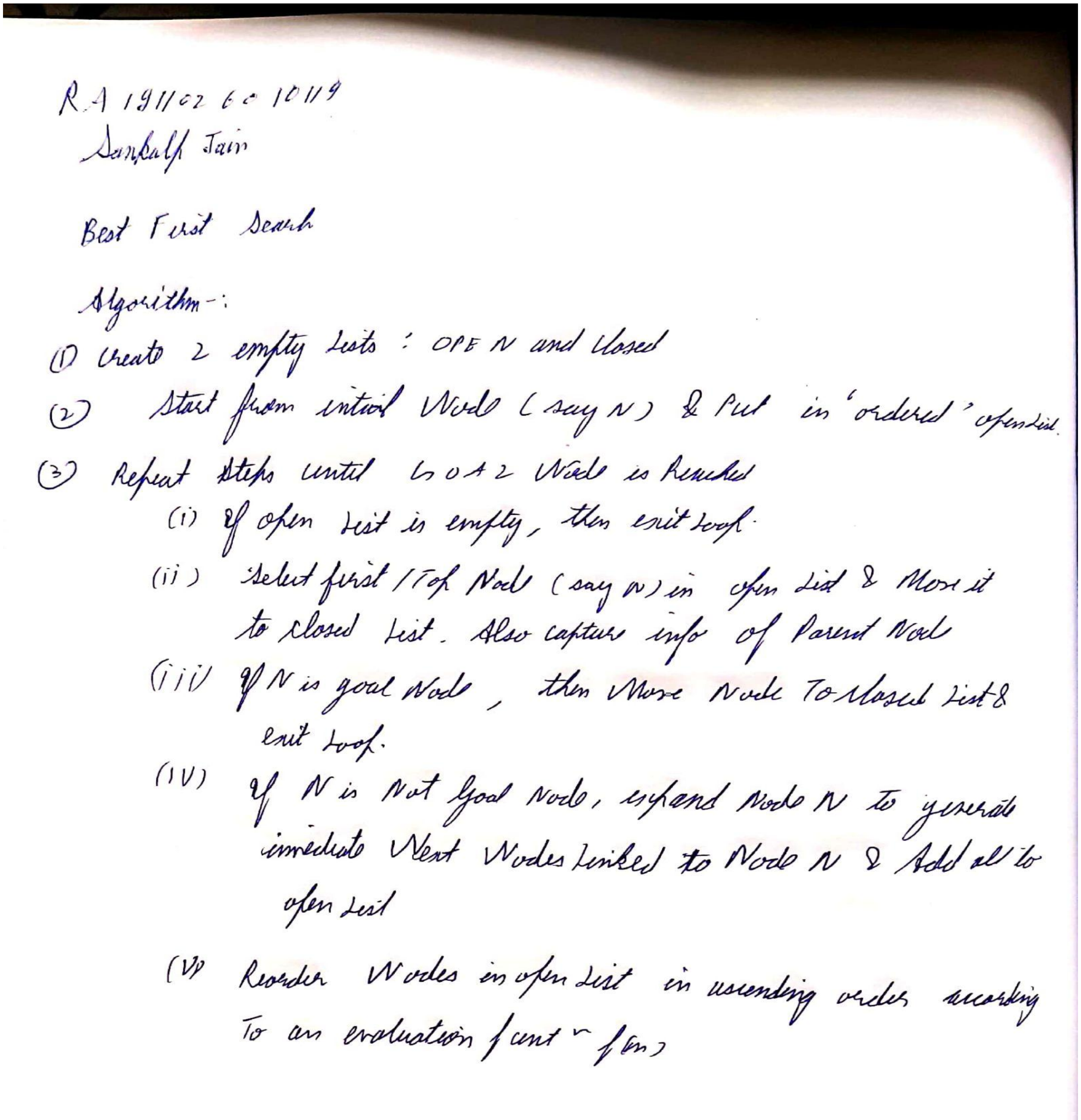
Exp 5-Best First Search

NAME- SANKALP JAIN
REG NO - RA1911026010119

AIM- To find the shortest path using BEST first search

ALGORITHM-



R.A 1911026010119
Sankalp Jain

Best First Search

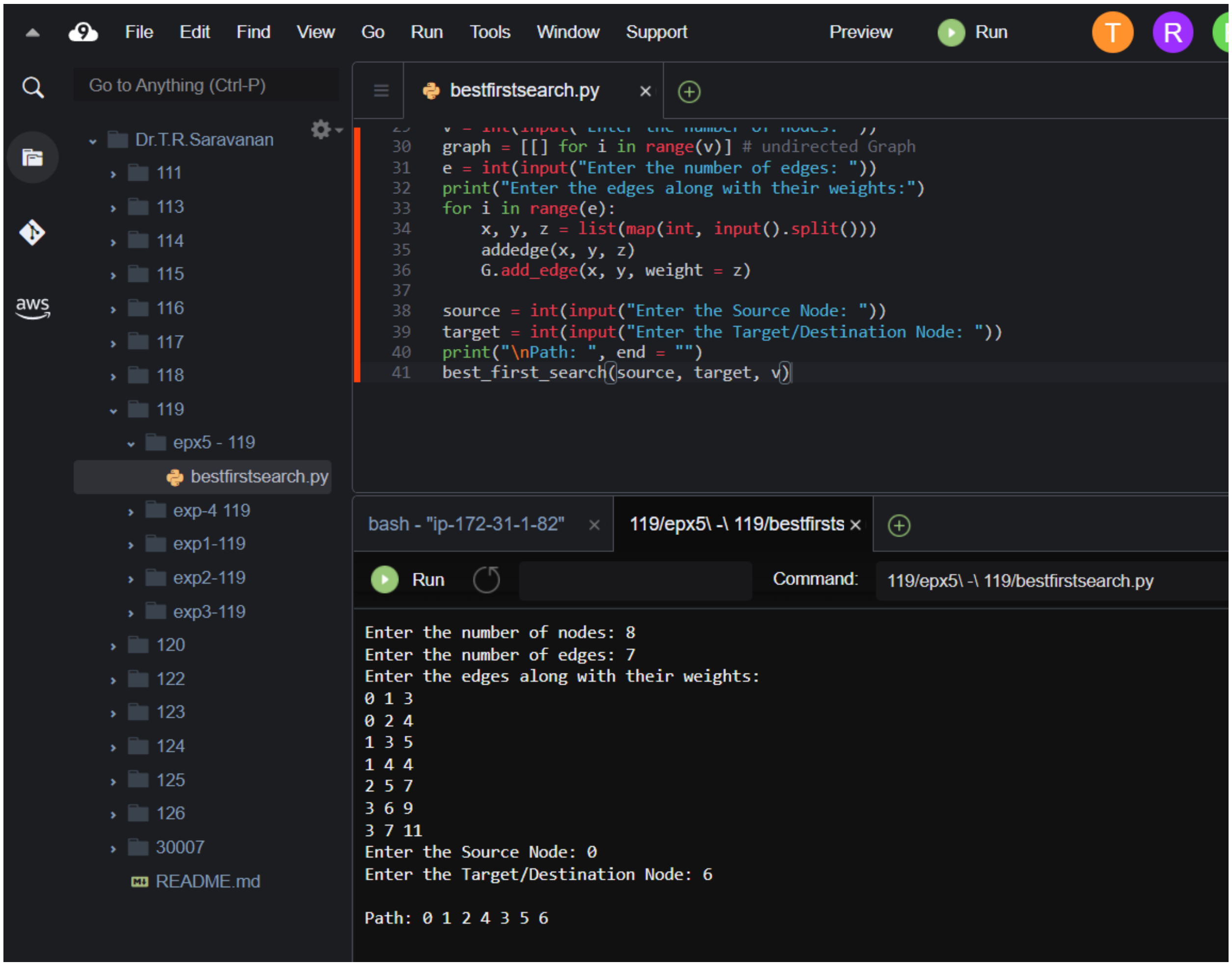
Algorithm-:

```
(1) Create 2 empty lists : OPEN and Closed
(2) Start from intial Node (say N) & Put in 'ordered' openlist.
(3) Repeat steps until GOAL Node is Reached
    (i) If open list is empty, then exit loop.
    (ii) Select first/Top Node (say N) in open list & Move it
         to closed list. Also capture info of Parent Node
    (iii) If N is goal Node, then Move Node To closed list &
          exit loop.
    (iv) If N is Not Goal Node, expand Node N to generate
         immediate Next Nodes linked to Node N & Add all to
         open list
    (v) Reorder Nodes in open list in ascending order according
        To an evaluation funct^n f(n)
```

CODE-

```
from queue import PriorityQueue
import matplotlib.pyplot as plt
import networkx as nx

# for implementing BFS | returns path having lowest cost
def best_first_search(source, target, n):
    visited = [0] * n
    visited[source] = True
    pq = PriorityQueue()
    pq.put((0, source))
    while pq.empty() == False:
        u = pq.get()[1]
        print(u, end=" ") # the path having lowest cost
        if u == target:
            break

        for v, c in graph[u]:
            if visited[v] == False:
                visited[v] = True
                pq.put((c, v))
    print()

# for adding edges to graph
def addedge(x, y, cost):
    graph[x].append((y, cost))
    graph[y].append((x, cost))

G = nx.Graph()
v = int(input("Enter the number of nodes: "))
graph = [[] for i in range(v)] # undirected Graph
e = int(input("Enter the number of edges: "))
print("Enter the edges along with their weights:")
for i in range(e):
    x, y, z = list(map(int, input().split()))
    addedge(x, y, z)
    G.add_edge(x, y, weight = z)

source = int(input("Enter the Source Node: "))
target = int(input("Enter the Target/Destination Node: "))
print("\nPath: ", end = "")
best_first_search(source, target, v)
```

OUTPUT-



```
v = int(input("Enter the number of nodes: "))
graph = [[] for i in range(v)] # undirected Graph
e = int(input("Enter the number of edges: "))
print("Enter the edges along with their weights:")
for i in range(e):
    x, y, z = list(map(int, input().split()))
    addedge(x, y, z)
    G.add_edge(x, y, weight = z)

source = int(input("Enter the Source Node: "))
target = int(input("Enter the Target/Destination Node: "))
print("\nPath: ", end = "")
best_first_search(source, target, v)
```

```
Enter the number of nodes: 8
Enter the number of edges: 7
Enter the edges along with their weights:
0 1 3
0 2 4
1 3 5
1 4 4
2 5 7
3 6 9
3 7 11
Enter the Source Node: 0
Enter the Target/Destination Node: 6

Path: 0 1 2 4 3 5 6
```

RESULT-Hence we successfully found the shortest path using BEST first search